EXHIBIT 7
(PART 2 OF 4)

SECTION 7

GRAPHIC DATA DISPLAY

Three types of graphic data input are possible on the 5216 Standard Firmware.

The first type is pixel-by-pixel entry using either the CUR instruction with the 'write dot' bit set, or one of the Write Pixel instructions (WPX, WPXF, WPXB). The CUR instruction enters pixel data in one pixel location at a time using the input mode specified in the MCW. (This is similar to the WRITE GRAPHIC (WRIT G) function key entry.) The Write Pixel instructions make use of Z axis data input as specified in either the WPX instruction, or the Foreground or Background Pixel Register. However, Memory Input Mode is ignored for the Write Pixel instructions.

In addition to the above mentioned pixel-by-pixel data entry instructions, the Block Transfer Mode (BXM) provides a high speed transfer of pixel data from the host to the 5216, when Pixel mode is selected.

The second type of graphic input is in blocks of data. In Load Graphic Elements (LGE), eight continguous picture elements of data are input in accordance with the currently selected Memory Input Mode. Word mode must be selected when LGE is used. Data is entered into all enabled memory channels.

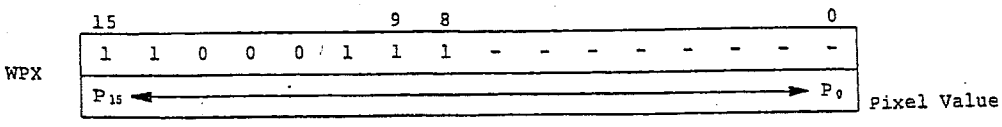
Blocks of graphic data may also be input using the BXM in graphic word mode. In this case, data is transferred at high speed from the host to the display computer. BXM in Word mode transfers 16 bits of contiguous data at a time to the graphics display. In this mode, after each word is transmitted the cursor will be automatically advanced by 16 bits.

In Word mode, Z axis bit patterns are selected by selecting major and minor channels. BXM Word mode transfers can be performed to one channel at a time or to more than one channel. (See the section on Block Transfer Mode.)

The third type of graphic input is conic generation. Conics are either vectors (straight lines) or circles. Arcs may be formed by generating circles within appropriate conic rectangular limits.

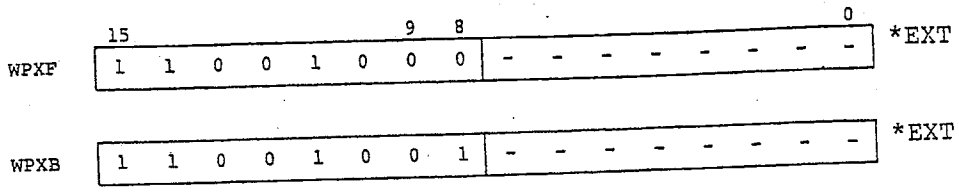
Write Pixel Values - WPX *EXT

AX209842



WPX causes the subsequent 16-bit word (P_0 - P_{15}) to be written into selected refresh memory modules as single picture element information, with the cursor advanced appropriately by the values ACA_x and ACA_y. Memory Input Mode is ignored.

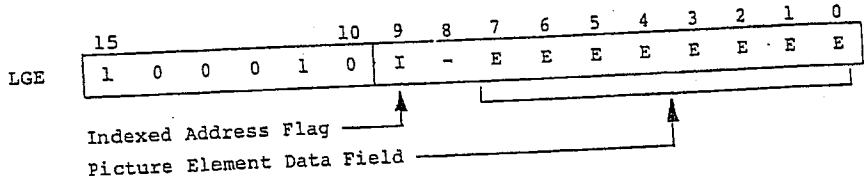
Write Pixel - Foreground and Background - WPXF, WPXB



WPXF specifies that the value contained in the Foreground Pixel Register be written into selected refresh memory modules as single picture element information. WPXB specifies the same action with the source being the Background Pixel Register.

The cursor is advanced following the execution of either instruction according to ACA_x and ACA_y. The Memory Input Mode ACA flag is ignored.

Load Graphic Elements - LGE



This instruction writes eight bits of picture element data into all selected memory channels, starting at the memory location addressed by the cursor X and Y coordinates. If bit 9 is set, the starting address for the memory write is $C_x + I_x$ and $C_y + I_y$, where C is the cursor coordinate and I denotes the index register content. Bit 0 of the LGE data element appears on the screen as the leftmost pixel of the eight pixels addressed.

After LGE has performed the memory write, the cursor is advanced to the right by the number of picture elements specified by ACA_x; however, the cursor advance for LGE does not depend on the ACA flag bit in the Mode Control Word.

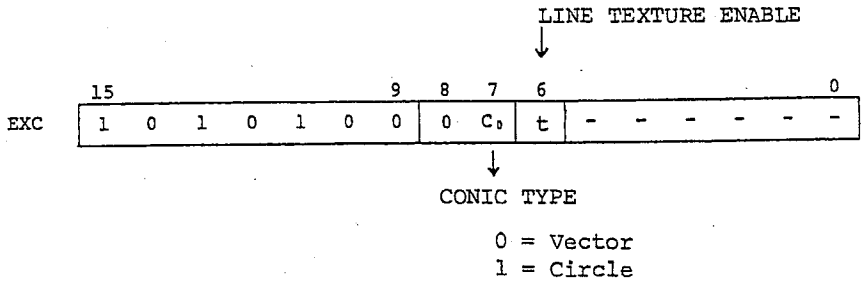
The programmer may enter from one to eight graphic elements and move the cursor the desired amount (0 to 31 picture elements) by

AX209843

the setting of ACA_x. The Graphic and Word display modes must be selected before LGE is executed.

Conic Generator (Circles and Vectors) - Execute Conic Instruction - EXC

Discrete approximations to vectors and circles are computed and written into all selected memory channels by the EXC instruction. The rectangular limits within which conics are written are specified by LCLL, LCLR, LCLT, and LCLB instructions. These limits are independent of the limits used to define areas to be scrolled, cleared, transmitted, or written into with BXM, LGE, or LAC instructions. If the Pixel mode is in effect, the foreground register defines the pixel value written into all selected channels (when in Replace or OR 1's mode). When in Erase mode, input data is defined by the background register.



Vectors

The start point of the desired vector is loaded into the cursor registers using LCX and LCY or CUR. The end point coordinate is loaded into the index registers using Load Index X (LIX) and Load Index Y (LIY). An EXC (vector) is then issued, causing the Display Generator to compute and write the necessary points to produce a line segment between the start and end points. The cursor is positioned at the end point when the vector is completed. Tip-to-tail vectors are drawn by respecifying the end point (by assigning a new index value) and issuing another EXC instruction. Random vectors are drawn by relocating both the cursor and index registers and issuing EXC.

Circles

The center point is loaded into the cursor registers. The radius of the desired circle is loaded into the X index register using LIX. Upon receiving an EXC (circle) instruction, the Display Generator will compute all of the points necessary to write a circle with the given center and radius. Upon completion of the circle, the cursor is left positioned at the center of the circle. Thus, to draw concentric circles, it is necessary only to reload X index to the desired radius, and repeat the EXC instruction.

Aspect Ratio Correction

Most raster scan CRT displays are rectangular, and each pixel is wider than it is longer. The ratio of pixel height to width is called the Aspect Ratio. For example, a refresh memory size of 1024 x 1024 displayed on a screen with a rectangular display area of 11¼ inches by 15 inches has an aspect ratio of 3:4.

Circles appear elliptical on such a display if the radius is measured in number of pixels. For example, on a display with an aspect ratio of 3:4, a circle with a radius of 10 pixels will appear 1.33 times wider than it is high.

AYDIN firmware automatically compensates for this difference between the height and width of pixels when the circle instruction is executed. This compensation is accomplished by using a variable number of pixels as the radius of the circle. For example, a circle with 15 pixels between its center and its right or left edge would have 20 pixels between the center of the circle and its top or bottom edge.

When drawing a circle, the programmer specifies the radius by giving the number of pixels (of radius) on the horizontal axis. The firmware will automatically draw the circle, based on the aspect ratio of the particular hardware display for which the firmware was configured.

Conic Limits

A special set of limits called Conic limits govern the input of conic display data. Conic limits are rectangular, and are specified by the four conic limit instructions: right, left, top, and bottom. New conic data will be entered onto the screen ONLY WITHIN CONIC LIMITS.

Load Conic Limits - LCLL, LCLR, LCLT, LCLB

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LEFT - LCLL	0	1	0	1	0	1	L	L	L	L	L	L	L	L	L	L
RIGHT - LCLR	0	1	0	1	1	1	R	R	R	R	R	R	R	R	R	R
TOP - LCLT	0	1	1	0	0	1	T	T	T	T	T	T	T	T	T	T
BOTTOM - LCLB	0	1	1	0	1	1	B	B	B	B	B	B	B	B	B	B

AX209845

The ten least significant bits are the binary values of the conic limits.

Conic limits are useful for drawing a circle so that only a desired arc will be displayed. However, the programmer should be careful to draw vectors or circles within conic limits. A vector or circle drawn outside conic limits will not return an error message, and the desired data will not be displayed.

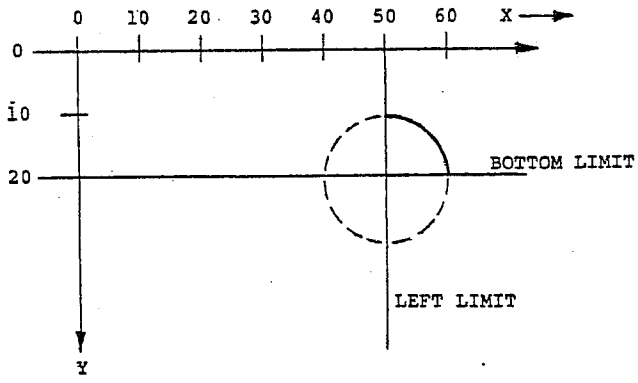
Conic limits are completely independent of the rectangular viewport discussed in Section 3.

Arcs of Conics

Since conics are drawn only within the conic limits, arcs can be generated by properly setting these limits before executing a conic instruction.

For example, suppose a circular are is to be drawn with the following parameters:

RADIUS - 10
CENTER - X = 50, Y = 20
START POINT - X = 50, Y = 10
STOP POINT - X = 60, Y = 20

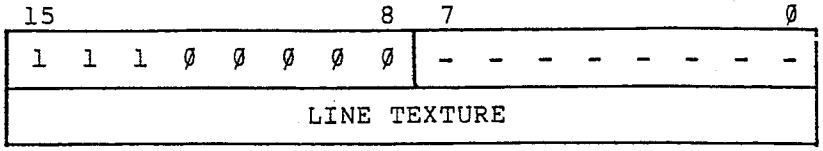


The following instruction sequence will cause this circular arc to be drawn: (The top limit is assumed to be less than Y = 10. The right limit is assumed to be greater than X = 60.)

Code	Description	
LCL, 50	Load left limit to 50	With conic limit bit set
LCB, 20	Load bottom limit to 20	
LCX, 50	Load cursor X to 50	Center
LCY, 20	Load cursor Y to 20	
LCY, 10	Load Index X to 10 - radius	
EXC, CIR	Execute circle	

Line Texture - STEXT (OPTIONAL)

STEXT



This instruction loads a line texture value given in the second word. Subsequently, whenever a textured line is drawn by setting bit 6 of the EXC command to 1, i.e., EXC = $A84Ø_H$. Each bit in the word corresponds to one bit in the vector. The texture value is repeated as often as necessary to finish the vector and is reset to the initial value loaded by this instruction. Currently, the first texture bit used is bit 14, the second is bit 13, etc..

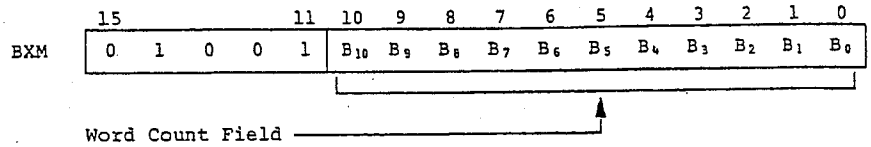
AX209847

SECTION 8

BLOCK TRANSFER MODE

The fastest way to download either alphanumeric or graphic data from the host to the 5216 Display Computer is by using the Direct Memory Access (DMA) ability provided by the Block Transfer Mode (BXM) instruction. After receiving just one BXM instruction, the 5216 is prepared to receive up to 2047 data words, or by using the Extended BXM (XBXM), up to 2^{24} - 1 data words. Cursor advance is managed automatically.

Block Transfer Mode - BXM



The data transmitted to the 5216 with the BXM can be either alphanumeric or graphic. The 5216 checks the state of the MCW to determine if data is to be processed as alphanumeric or graphic. The BMX instruction word is the same in both cases.

Graphic data can be processed in two different ways, depending on the status of MCW bit 1 (Word/Pixel mode). In Word mode, each BXM graphic data word addresses 16 X,Y locations, and in Pixel mode, each BXM graphic data word addresses only one X,Y location.

Caution must be exercised to ensure that the BXM instruction is followed by the number of data words specified in the word count field. Once the BXM instruction is received, the number of words specified in the word count field will be treated as data words. The three ways the BXM instruction can be used are discussed below.

Block Transfer Mode-Alphanumeric

In the Alphanumeric mode, the data words following the BXM instruction are interpreted by the 5216 as two eight-bit alphanumeric codes.

The characters transferred to the 5216 in this mode use the font size, cursor advance, Memory Input Mode, and unpacking order

AX209848

specified in the MCW. Characters are displayed at the current cursor position and advanced according to the same rules by which the cursor is advanced after each Load Alphanumeric Character (LAC) instruction.

When the side boundary of the rectangular viewport is encountered, the cursor is advanced up or down, according to the cursor advance currently in effect, and then restored to the opposite side of the viewport.

BXM Alphanumeric may be operated in either Pixel or Word mode. Also see Section 4 (MCW) for the data unpacking order of bit 2.

Block Transfer Mode - Graphic

Word mode

In the Graphic word mode, the data words which follow the BXM instruction are used as words of pixel data, each word representing 16 pixels. For each bit = 1 in the data word, the corresponding pixel in refresh memory is set to all 1's. For each bit = 0 in the data word, the corresponding pixel in refresh memory is set to all 0's. Of course, only selected channels are affected by this data input.

The data words are written into 16 consecutive pixel locations, starting from the cursor location and incrementing to the right. When the right viewport limit is encountered, the Y coordinate is incremented and the cursor is restored to the left of the viewport rectangle.

To operate in BXM Graphic Word Mode, the programmer must set the graphic and word bits equal to 1 in the MCW.

Viewport limits are automatically forced to word boundaries, which are defined on the left when the low four bits = 0, and on the right when the low four bits = 1. Previously specified viewport limits are restored after the transfer is completed. The difference between the left and right limits determines the number of 16-bit words per line:

$$(RL - LL + 1) / 16 = \text{number of words/line}$$

Two examples of the areas filled by a Graphic Word BXM are shown in figure 8-1.

To rapidly create images containing several different pixel values, the programmer can send graphic data to the 5216, selecting one channel at a time and varying the values on each channel.

Complex images can be created on the display monitor using keyboard function input or digitizing hardware. Then, using the Transmit Selected Channels (TSC) instruction, the displayed image can be saved in data words to the host. Using the BXM instruction,

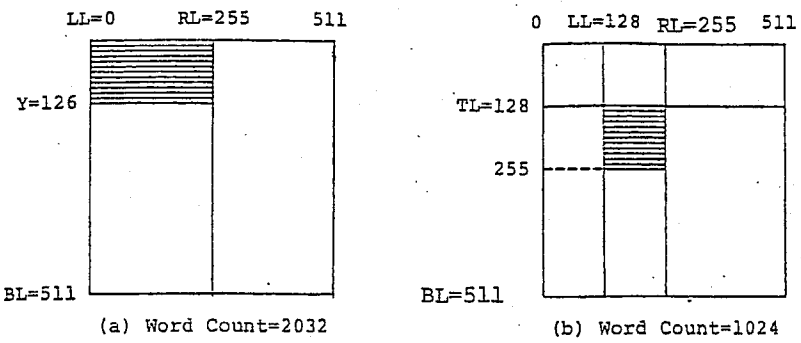


Figure 8-1. Areas Filled by a Graphic BXM

these data words can be used to retransmit the image to the display.

Pixel Mode

Each data word following the BXM instruction in Graphic Pixel mode contains 16 bits of pixel data in the Z direction, to be written to a single pixel location. After each pixel is written, the cursor is advanced by one pixel, and the direction of advance is determined by special rules for BXM graphic pixel transfer. These rules use the value of the Adjustable Cursor Advance (ACA).

A positive ACA_x value will cause the cursor to move from left to right; a negative value will cause it to move from right to left. A positive ACA_y value will cause the cursor to move from top to bottom; a negative value will cause it to move from bottom to top. If the magnitude of the ACA_x value is greater than or equal to the ACA_y value, the BXM instruction will enter data in horizontal lines. If the magnitude of ACA_y is greater than the magnitude of ACA_x, the BXM will enter the data in vertical lines.

For example:

$$ACA_x = -1 \qquad ACA_y = 3$$

In this example, the data is entered in vertical lines moving from right to left. The data is loaded starting at the current cursor location and proceeding to the rectangular limit (i.e., for vertical lines moving from top to bottom, the Y cursor address would be incremented until the bottom rectangular limit is encountered) at which point the cursor is set to the opposite rectangular limit and moved over to the next line. In the example given, the Y cursor is set to the top limit and the X cursor

is decremented. When the X cursor reaches the left rectangular limit, it is reset to the right rectangular limit.

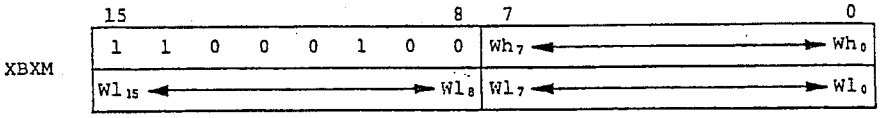
For graphic Pixel BXMs, the rectangular limits may be any pixel address, as opposed to the graphic Word mode, in which the limits are altered to word boundaries. Graphic Pixel mode BXM uses Replace data mode only, regardless of how the MCW is set.

There is complete control over pixel values, with the disadvantage that one word must be transferred for each pixel. (Each data word in this mode transfers data for 16 pixels.) Special cursor advance features can be used to rotate images 90, 180, or 270 degrees by changing the ACA values.

To use this mode, the graphic and pixel bits in the MCW must be set.

The Transmit Picture Elements (TPX) instruction is used to transmit data from the 5216 display to the host in a form which is compatible with the BXM graphic Pixel mode.

Extended Block Transfer - XBXM *EXT



Block transfers of up to 2^{24} - 1 words may be specified by XBXM, a 16-bit two-word instruction sequence. The high order byte of the first word is the XBXM OP-code. The low order byte of the first word specifies the eight most significant bits of the 24-word count. The second word in the instruction sequence specifies the 16 least significant bits of the word

AX209851

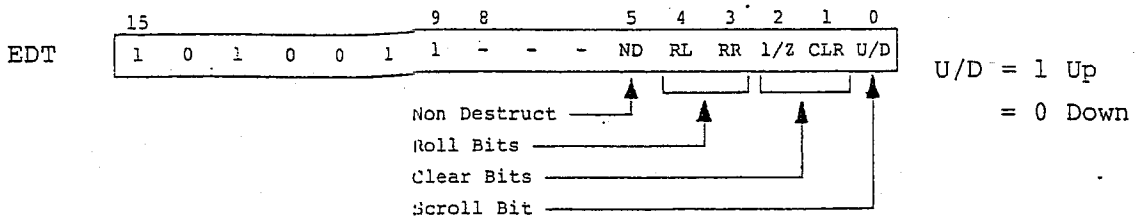
SECTION 9

DISPLAY EDIT FUNCTIONS

The Edit instruction (EDT) provides for scrolling (up or down), rolling (right or left), and clearing (to 0's or 1's). These operations are performed only within the rectangular window limits. Roll and scroll may be performed destructively or with wrap around. An Extended Edit Instruction (XEDT) enables the programmer to specify the number of pixels to be scrolled or rolled by a single command.

Other fucntions described in this section are Zoom, Fill, and Copy. These instructions are identical to the corresponding function keys included on the keyboard, and are used to affect data which is already displayed.

Scrolling, Rolling, Clearing - Edit Instruction EDT



1/Z ⇒ ONEs or Zeros

ND = 1 is non-destructive
ND = 0 is destructive

EDT provides for clearing any combination of selected channels simultaneously, either to all 1's or all 0's. In addition, all data within the rectangular limits may be scrolled up or down, left or right, or one raster line with the scroll/roll instruction. The three least significant bits of this instruction are decoded as follows:

Bit No. 2 1/Z	1 CLR	0 SCR	Meaning
0	0	0	Scroll Down
0	0	1	Scroll Up
0	1	0	Clear to Zeros
1	1	0	Clear to Ones

Scroll

Zeros in bits 1, 2, 3, and 4 of EDT define the scroll instruction, which causes the displayed data in selected channels to move up or down one vertical element. The scroll operation is performed only on data within previously defined rectangular limits. Data in each selected channel is scrolled. The execution time for the scroll instruction depends on the values of the rectangular limits.

Roll

Bits 3 and 4 define the roll instruction. They are defined as follows:

Bit 4	3	Meaning
0	1	Roll Right
1	0	Roll Left
1	1	Danger

A one in bit 3 or 4 causes displayed data in selected channels to move right or left one horizontal element within rectangular limits. A one in both bits 3 and 4 specifies a conflicting operation and will cause an error signal.

Clear

Bit 1 of EDT specifies that a memory clear operation is to be performed. When bit 1 = 1, all selected memory channels are cleared to zeros if bit 2 = 0 or are cleared to ones if bit 2 = 1. Selected channels are simultaneously cleared only within the current values of the rectangular limits.

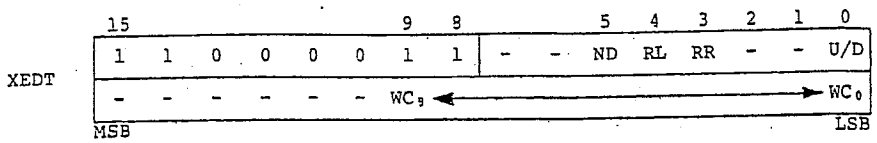
Destruct/Non Destruct 1-ND

When bit 5 is set to 1, any data which is rolled or scrolled beyond the limits of the rectangular window will wrap around.

When bit 5 = 0, data which is rolled or scrolled beyond the currently selected window will be lost.

AX209853

Extended Edit - XEDT *EXT



U/D = 1 Up
= 0 Down

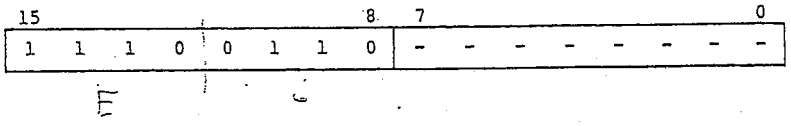
The XEDT instruction provides for the scrolling or rolling of all data within rectangular limits, 0 to 1023 raster lines or columns at a time.

Bits 0, 3, 4, and 5 of the first word of the two-word XEDT instruction are decoded identically to bits 0, 3, 4, and 5 of the EDT instruction.

Bits 1 and 2 of the first word of XEDT are ignored as clearing is undefined for XEDT.

Bits 0 through 9 (WC_0-WC_9) of the second word in the instruction sequence specify the number of raster lines/columns to be scrolled/rolled. A maximum of 1023 lines/columns may be specified.

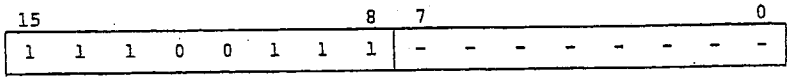
Zoom *EXT



This instruction causes a 2 to 1 magnification of data within the rectangular window. Starting from the center of the window all distances from pixel to center are doubled. Any data pushed outside the window by this function will be lost.

This instruction is identical to the Zoom function key.

Fill *EXT



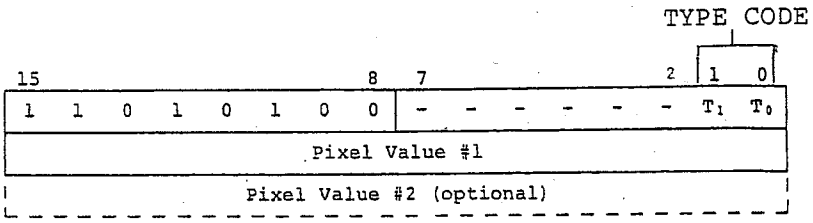
AX209854

This instruction is used to fill an area with a new pixel value. The area to be filled is selected by locating the cursor over any pixel in the area. The area is bounded by any pixel which does not equal the pixel value over which the cursor was located when the instruction was executed. The area to be filled is also bounded by the rectangular window limits.

The new Pixel value with which the area is to be filled is the value in the Foreground Pixel Register (FPR).

In Pixel mode, the foreground pixel value is used no matter what Memory Input Mode is selected. In Word mode, 1's will be entered in all selected memory planes within the selected area.

Copy *EXT



This function copies pixel data from one rectangular area of the screen and places it into another. The area to be copied to is defined by the conic limits. The area to be copied from has the same size as the conic limits, and is located on the screen using the cursor position as its center. No change is made to the area which is copied; rectangular viewport does not affect the Copy instruction.

The Copy instruction will be followed by one or two fields of data depending on the setting of bits 0 and 1 in word 1. The Copy function is instructed as to which pixels are to be copied by specifying either a particular pixel value or a range of pixel values.

Since the Copy function selectively copies pixels, it may be used to copy a window onto itself by locating the cursor in the center of the conic window.

The value of the pixels to be copied is determined by the type code, pixel value #1, and optional pixel value #2. If type code = 0, only pixels which when AND'ed with pixel value #1 and yield a non-zero result are copied. If type code = 1, only pixels equal to pixel value #1 are copied. If type code = 2, only pixels with values lying within the range defined by pixel value #1 and pixel value #2 are copied. If type code = 3, only pixels with values not within the range defined by pixel value #1 and pixel value #2 are copied.

In OR 1's mode and Erase 1's mode, pixels in the conic window which weren't copied into are not changed. In Replace data mode and Replace Reverse data mode, pixels in the conic window which weren't copied into are set to zero if the MCW is set for Word mode, or they are set to the Background Pixel Register (BPR) value if the MCW is set for Pixel mode.

T_1	T_0	Number of Parameter Words	Will Copy
0	0	1	non-zero after AND
0	1	1	equals
1	0	2	in range
1	1	2	not in range

Memory Input Mode	Pixel/Word Mode	Pixels not copied into will be
OR 1's	Don't Care	Not Changed
ERASE 1's	Don't Care	Not Changed
Replace Normal or Reverse	Word	Set to Zero
Replace Normal or Reverse	Pixel	Set to Background Pixel Register value

SECTION 10

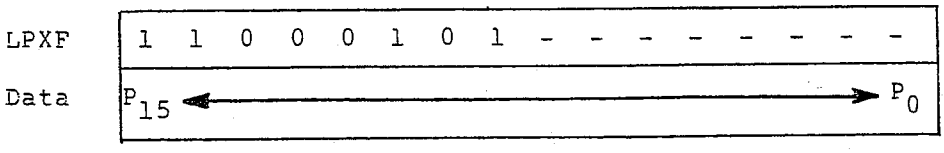
PIXEL VALUES

Each picture element, or "pixel," in refresh memory has a numerical value. This is sometimes called the Z value because it is the third dimension of the pixel. The first two dimensions are the X and Y values of the pixel, which describe its location on the two-dimensional coordinate space of the screen.

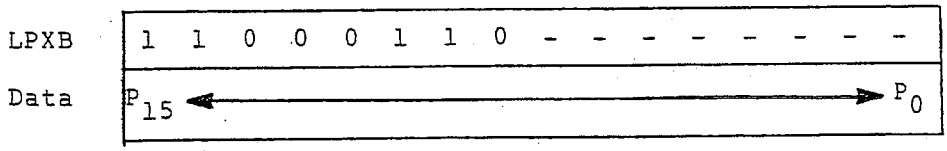
This pixel, or Z, value may be assigned to pixels in alphanumeric or graphic display by selecting Pixel mode in the Mode Control Word (MCW), and assigning the desired pixel values to the Foreground and Background Pixel Registers (FPR and BPR, respectively). The values stored in the FPR and BPR are then used by the 5216 Standard Firmware to draw lines, circles, and alphanumeric data according to the rules associated with the currently selected Memory Input Mode.

For example, in Replace Normal mode, the value in the FPR is used for each pixel in the character display and the value in the BPR is used for each pixel in the character background block. (See Section 4 for a complete description of the input rules.)

Load Foreground Pixel Register - LPXF *EXT



Load Background Pixel Register - LPXB *EXT



Pixel values act as bit maps to determine which memory channel will have its bit set at the X,Y location being addressed. For example, a pixel value of 00F0 hex will set memory bits in the fifth, sixth, seventh, and eighth memory planes only.

The programmer must be aware of the memory channels available in his own system, and the channels currently selected. No error condition is created by writing to nonexistent or unselected memory channels.

SECTION 11

CHANNEL SELECTION AND MASKING

The 5216 Standard Firmware can support system configurations with up to 16 memory channels and each channel can be subdivided into halves and quarters. The memory channels are selected by instructions referring to "major Channels" and the subdivisions of each board are referred to as "minor Channels". The size of each major Channel and the type and size of the minor Channels will depend on specific system configuration.

Data displayed on the screen is output from all memory channels, major and minor, available in the system and is not affected by channel selection. Selecting or deselecting channels only affects the acceptance of input data.

Masking channels is not a necessary procedure for deselecting channels. Masking is used for protection and will cause an error if a selected channel is masked or a masked channel is selected. Channel selection is one means of selecting picture color.

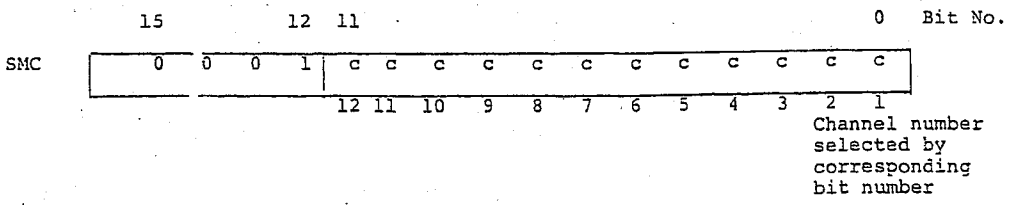
When Word mode is selected, data is written into every selected channel. Changes in selected channels will control the input value of pixel data, but will not affect data already stored in refresh memory.

The output of the picture element data is used as an address to reference the current lookup table value. For example, if the first three channels are selected when display data is entered, and if the data is entered in the three planes selected (for example, in Word mode, Replace Normal) then the pixels affected by this data will have the low three bits set. These low three bits will then reference relative address 7 decimal of the lookup table.

In some hardware configurations, channel selection is used to direct data for one picture to a subset of memory channels; for example, the low four planes and another picture is directed to another subset of memory channels, i.e., the high four planes. In this configuration, each subset of memory channels is wired to a separate VID card.

In this example, each video card drives a monitor; one video card is wired to the low four memory channels, and the other video card is wired to the high four memory channels. The 5216 can be configured with as many as four video cards using as many as 16 memory channels, each divided into 1, 2, or 4 minor channels.

Select Major Channel - SMC



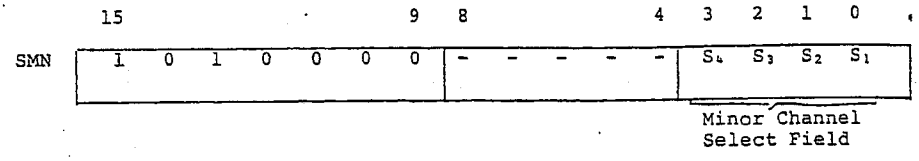
The Select Major Channel instruction contains a 12-bit channel select field. Each bit of the channel select field (bits 0-11) enables the inputs of one of up to twelve refresh memory channels. The SMC instruction is used to select any combination of memory channels for use in subsequent instructions.

The low bit of SMC corresponds to the low order memory plane in the system. Any bit set (1) will enable the corresponding channel; any bit 0 will disable the corresponding channel.

After SMC, any data written into the refresh memory (characters, vectors, etc.) in the Word mode (See MCW) is entered at a particular location in each of the selected channels. A channel which is not selected will receive no further input data, but its previous contents will continue to be displayed. SMC is used in conjunction with MMC to provide memory protection (described in next section). SMC may be issued many times to construct a complex picture.

For systems with more than 12 major memory channels, use the Extended Select Major Channels (XSMC) instruction.

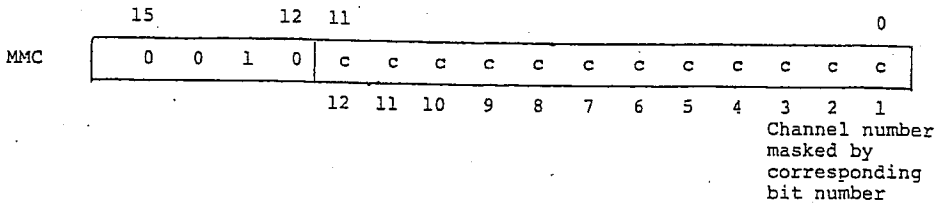
Select Minor Channels - SMN



AX209859

The Select Minor Channel instruction is used only with channel configurations that sub-divide a memory board into 2 or 4 logically separate arrays. In a system with four minor channels, the minor channel select field functions analogous to the SMC instruction, with each bit specifying a particular minor channel. In those configurations where a memory is divided into 2 minor channels, minor channel 1 is selected by $(S_4S_3S_2S_1) = (0001)$, minor channel 2 by $(S_4S_3S_2S_1) = (0100)$, and both may be simultaneously selected by $(S_4S_3S_2S_1) = (0101)$.

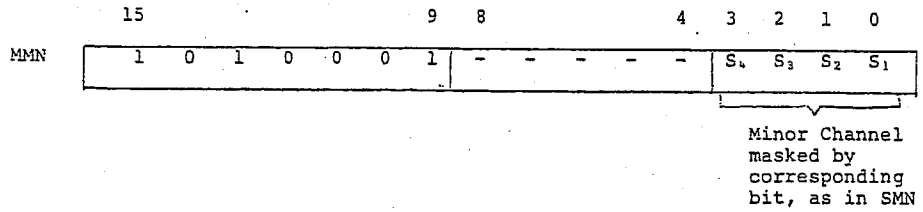
Mask Major Channel - MMC



MMC causes the memory channel inputs indicated in the channel mask field to be disabled, inhibiting further entry of data into the corresponding memory channel. This instruction is mainly for use by an executive routine for memory protection.

To ensure consistency in programs involving protected channels, only unselected channels may be masked, and only unmasked channels may be selected. Selecting a masked channel causes an error message to be transmitted to the host computer. To clear the error, deselect the offending channels and select only unmasked channels thereafter. Similarly, the error caused by masking a selected channel is cleared by unmasking the channel.

Mask Minor Channels - MMN

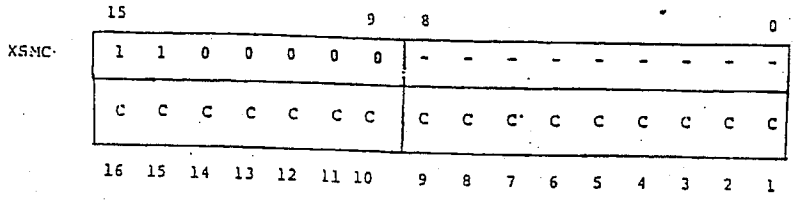


The Mask Minor Channel instruction provides for protecting minor channel information in a manner similar to that described in MMC.

AX209860

Extended Channel Selection and Masking - XSMC, XMMC

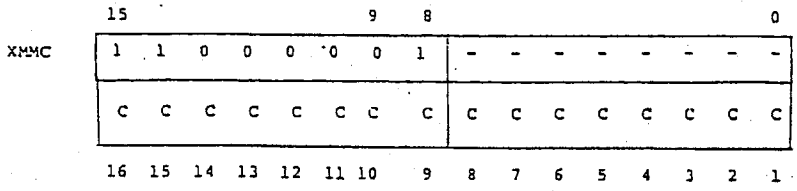
Extended Select Major Channel - XSMC *EXT



Bit Map

The XSMC instruction defines the next word to be a channel select of up to 16 channels. (Channels 1 to 12 are the same as SMC 1 to 12.)

Extended Mask Major Channel - XMMC *EXT



Bit Map

XMMC defines a channel mask of up to 16 channels.

Except for the two-word sequence and extra four-channel addressing bits, these instructions are identical to SMC and MMC.

SECTION 12

VIDEO CARDS - VID-004 AND VID-101

The video lookup table is a user programmable means of translating pixel data in refresh memory to color values on a Red Green Blue (RGB) monitor.

Each pixel value in refresh memory is cycled through the lookup table and the value loaded in the lookup table at the address specified by the lixel value is then output to the Digital-to-Analog Converters (DACs) for red, green and blue.

This provides a powerful means of entering color data into the refresh cycle without changing pixel data. For example, the displayed color of all pixels with a certain value can be changed with one instruction by changing the lookup table value for that pixel value. A simple case of this technique would be to load the zeroth position of the lookup table with the color value for red. Then, all pixels with a zero value would appear red on the screen.

A more sophisticated use of the video lookup table would be to assign serial output data from one subset of memory channels to one set of color values, and serial output data from another nonoverlapping subset of memory channels to another set of color values.

Then, whenever data is entered in both sets of memory channels, the pixel value may be referenced to a video lookup table value which is the same color value as would have been referenced if only the first subset had contained data. For example, if subset 1 contains the high four channels and subset 2 contains the low four channels, and if C0 hex (high two channels set) looks up red and if 03 hex (low two channels set) looks up blue, then the lookup table can be loaded so that C3 hex also looks up red. Therefore, the programmer has effectively established a priority of memory planes so that data in the high four planes overrides data in the low four planes.

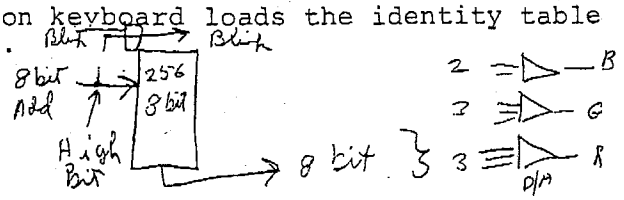
Loading the video tables in certain ways can also be used to make pictures or sections of pictures visible or invisible.

Two different video cards are available with the 5216. The specific lookup table values for each color depends on the VID card type in a specific system as does the size of the table.

The LUT #1 key on the function keyboard loads the identity table into the zeroth VID-004 card.

VID-004 VIDEO CARD

Description



The VID-004 card provides one 256 x 8 lookup table with a blink option and one Status Word to set overlay priorities. The 8-bit lookup values drive three on-board Digital-to-Analog Converters which in turn are used to drive the red, green, blue guns of a cathode ray tube

monitor. The low three bits drive the red gun, the next three bits drive the green gun and the high two bits drive the blue gun.

In addition to controlling priority overlay, the Status Word can enable and disable display of data generated by the Alphanumeric Channel Set (ANCS).

Blink Option

The high address bit is (optionally) AND'ed with a blink oscillator so the displayed color value of a pixel with the most significant bit set high oscillates between the two color values loaded at the address with and without the most significant bit set.

Status Word and Overlay Priority

The Status Word is used to control the priority options of the VID-004 card. For every pixel output by refresh memory, data of lower priority will be forced to zero if non-zero data of higher priority exists for that pixel.

The priority modes of the VID-004 card are designed for a typical system with eight graphics memory planes and an Alphanumeric channel, and are selected by four bits.

When all four bits of the Status Word equal 1, the Alphanumeric channel takes highest priority. Second priority is given to the eight graphics planes. When the first, second and third bits are 1 and the fourth bit is 0, the Alphanumeric channel takes highest priority. Low priority is given to the high five graphics planes.

An explanation of the four bits of the Status Word follows.

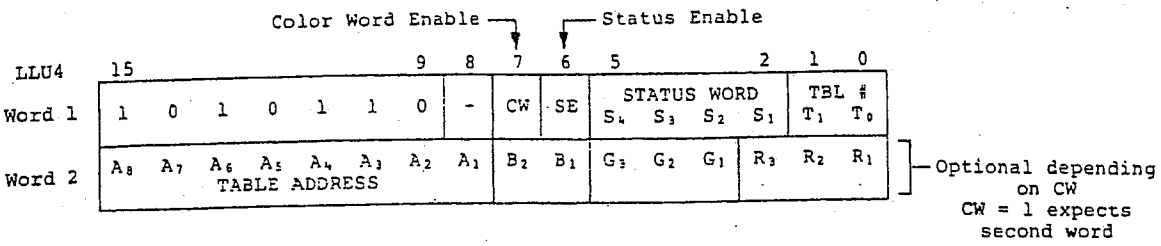
S1 When this bit is 1, the first three memory planes are enabled. When this bit is 0, the low three memory planes are disabled. All output from disabled planes will be zero.

S2 When this bit is 1, the high order five memory planes are enabled. When this bit is 0, the high five graphics memory planes are disabled.

S3 The Alphanumeric channel is enabled when this bit is 1 and disabled when this bit is 0. (This does not apply for systems where the A/N channel is not wired through the VID card.)

S4 When this bit is 1, all eight graphics planes will have equal priority. When this bit is 0, the low three graphics memory planes will have lower priority than the high five graphics planes. (S1 and S2 must be set to 1 to enable the planes.) Bit S4 has no affect on the priority of the A/N channel.

Load Lookup Table 004 - LLU4 *EXT



The least significant two bits of the first word select the VID-004 lookup table to be programmed. Each table number corresponds to the number of the VID cards being addressed. A system may contain up to four VID cards. To address more than four VID-4 cards, the Extended Load Lookup Table (XLLU4) instruction must be used. For a system with only one VID card, the table number is always 0.

Bits 2 through 5 specify the Status Word, which controls operation of the overlay capability in the VID-4 card.

Bit 2 = Graphics Enable (low 3 channels)
Bit 3 = Overlay Enable (high 5 channels)
Bit 4 = A/N Enable
Bit 5 = 8-Bit Operation Enable

For normal eight memory plane graphics display with no overlay priority, bits 2, 3, and 5 must be set to 1.

Bit 6 of the first word is the Status Enable bit. The Status Word (bits 2 to 5) is ignored if bit 6 is 0 and the Status Word is entered to the VID card if bit 6 is set to 1.

Bit 7 of the first word is the Color Word enable bit. If this bit is set to 0, LLU4 is a one-word command (used for changing the Status Word). If bit 7 is set to 1, the next word sent from the host is interpreted as color data.

The second word contains the color data (valid only if bit 7 of the first word is set to 1). The low three bits select the intensity of the red gun, from no intensity (000) to high intensity (111). The next three bits select the intensity of the green gun and bits 6 and 7 select the intensity of the blue gun.

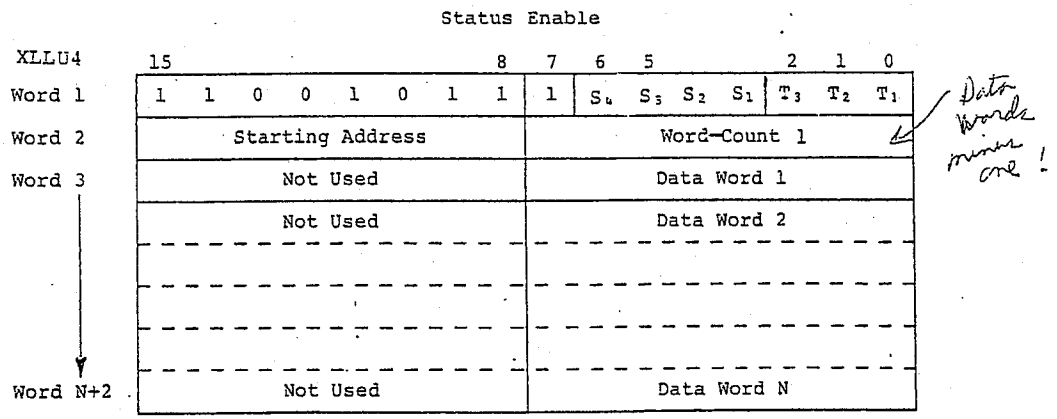
The address of the color table data is given in bits 8 to 15 (relative address from 0 to 256).

AX209864

This instruction is not the same as the LUT #1 and LUT #2 function keys on the 5116 keyboard. Using those function keys loads predefined values into the lookup table in the first VID-004 card.

To load more than one lookup value per instruction, use XLLU4 (Extended Load Lookup Table).

Extended Load Lookup Table - XLLU4 *EXT

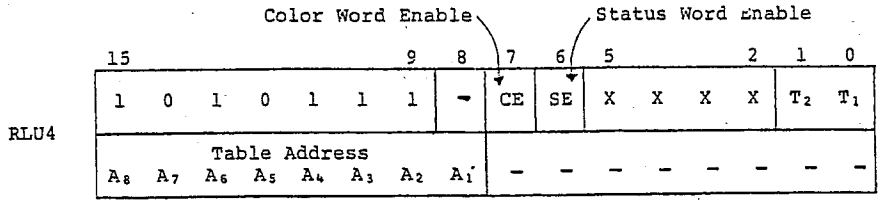


This instruction is used to load lookup table data to the VID-004 card numbered in the first three bits of word 1. Up to 8 VID-4 cards may be addressed w/ this instruction. For a system with only one VID card, these bits should be 0. Bits 3 through 6 give the Status Word as described for the LLU4 instruction. When bit 7 is reset to 0 the Status Word is ignored. The LLU4 instruction should be used to change the Status Word if no other color data is to be entered.

The low eight bits of word 2 contain the number of data words to be entered minus one. The number of words can be from 1 to 256, so that the number entered in this word can be from 0 to 256.

The high eight bits of word 2 contain the starting address in the lookup table where the sequence of data is to be entered; this can be from 0 to 255. Since the lookup table contains 256 words, the starting address plus the word count should not exceed 256.

Read Lookup Table 004 - RLU4



AX209865

RLU4 causes the selected lookup table values to be transmitted from the 5216 to the host. The first three bits of word 1 specify the VID card number from 0 to 3. Up to eight VID-004 cards may be supported by this firmware. RLU4 causes a one or two word LLU4 command to be sent to the host depending on whether bit 8 of the first word is set. If bit 8 equals 1, the second word that is sent to the 5216 will contain the address of the color value to be read from the lookup table specified in the first two bits of word 1. In this case, the LLU4 which is sent will contain two words.

If bit 7 is reset to 0, RLU4 is a one word command band the LLU4 sent to the host is also one word. Bits 0 and 1 of word #1 of RLU4 specify the color lookup table to be read. Bit 6 of word 1 is copied into bit 6 of the first word of the LLU4 command sent to the host. Bits 2 through 5 of the first word of the LLU4 command contain the status bits of the lookup table being read. (see Appendix A for transmit format.)

Extended Read Lookup Table 004 - XRUL4 *EXT

Status Enable

XRLU4	15							8	7	6	5			2	1	0
Word 1	1	1	0	0	1	1	0	0	1	S_4	S_3	S_2	S_1	T_3	T_1	T_0
Word 2	Starting Address A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Word Count 1 C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

The XRLU4 instruction causes transmission of the lookup table data from the VID card numbered in the first three bits of word 1.

The function of Status Word and the status word enable bit are the same as in the LLU4 instruction. The second word of the XRLU4 instruction contins the number of words to be read minus one and the address of the first word.

The second word of the reply message is the XLLU4 instruction and the third word is identical to the second word of the XRLU4 instruction. The status bits (3 through 6) of the reply message are copied from the lookup table status register. The status word enable bit (bit 7) of the reply message is copied from the status enable bit of the XRLU4 command.

AX209866

VID-101 VIDEO CARD

Introduction

The VID-101 has a 2048 x 12 main lookup table with 11 address bits and 12 data bits. This lookup table can also be configured as 4096 x 6.

There are three summing RAMs for red, green, and blue gamma correction or overlay and there are two monochrome summing RAMs, each having eight address lines in and four datalines out. Four DACs on board provide red, green, and blue analog output from four bits input each and a monochrome DAC with eight bits in. The Command Status Register (CSR) selects the addressing mode of the main lookup table (4096 x 6 or 2048 x 12). Another bit in the CSR selects a blink option which is AND'ed with the low order address bit of the main lookup table.

Provision is made for four cursor channels. The logical OR of two cursors is summed into the video signal after the summing RAMs. The logical OR of the other pair of cursor channels is input into the summing RAMs. There are two blink oscillators. Blink oscillator 1 is AND'ed with the output from memory channel 15. The output of blink oscillator 2 is AND'ed with the blink bit of the CSR. The result is inverted and AND'ed with the low address bit of the main lookup table. Another output of blink oscillator 2 is AND'ed with the logical OR of MEM 12 and MEM 13.

Provision is made on the VID-101 card to overlay the output from the Alphanumeric Channel Set. The overlay priorities are:

high; The OR'ed result of cursor pair 0 and 1.

middle; ANCS output

low; refresh logic

The lookup table, summing RAMs and CSR on the VID-101 card may be loaded by using the Load Cache Buffer command (LCB). See Section 14 for details of the Load and Read Cache Buffer instructions.

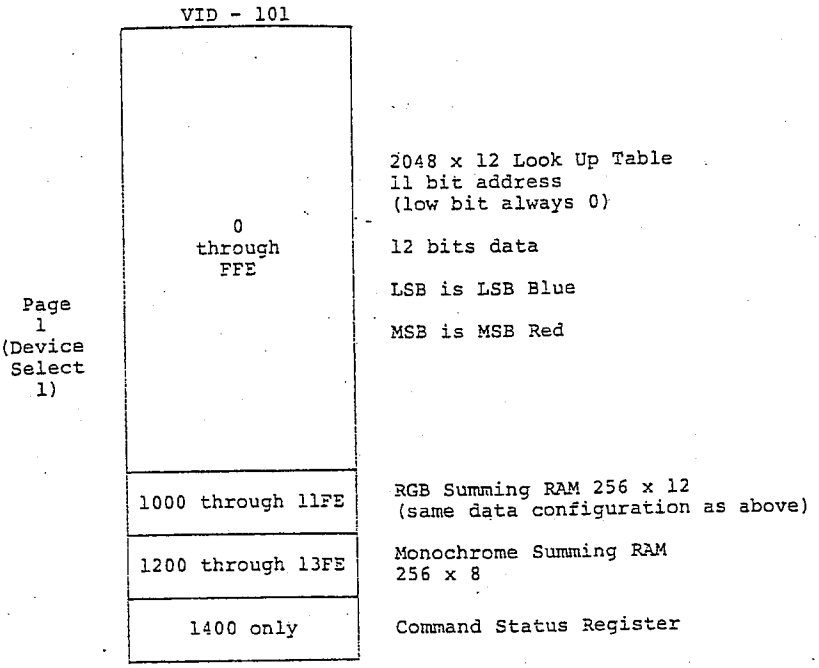
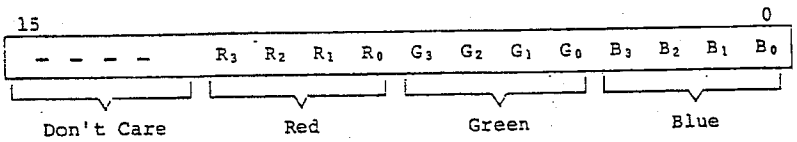


Figure 12-1. Memory Map of the VID-101 Card

Loading the Main Lookup Table - 004

The main lookup table is loaded at address 0 through address FFE on word boundaries (low address bit always 0), using the Load Cache Buffer (LCB) instruction (see Section 14). The data format is:



AX209868

The Device Select (page) number to be used for loading the VID-101 color tables is 1. The serial output of the refresh memory channels is shifted to the left one place and a zero is introduced at the low position so, for example, if memory plane 0 is high and all other planes are low, address 002 of the main lookup table will be referenced.

When the main lookup table is in 4096 X 6 mode (bit 0 of the CSR set high), data is loaded in the lookup table in exactly the same way. However, in this mode, MEM 11 low enables LUT output bits 2, 3, 6, 7, 10, 11 (RGB A2, A3 and Monochrome A2, A3) and MEM 11 high enables LUT output bits 0, 1, 4, 5, 8, 9 (RGB A0, A1 and Monochrome A0, A1). MEM 11 is an extra input which is not used in 2048 X 12 mode.

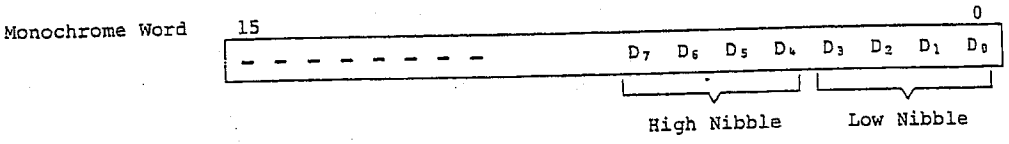
Loading the Color Summing RAMs

The color summing RAMs are located at addresses 1000 hex through 11FE hex (memory page 2). These summing RAMs are loaded using the Load Cache Buffer command on memory page 1, at these addresses. The three 4-bit color summing RAMs are loaded as if they are one 12-bit table. The color data word has the same format as the color data word for the main lookup table with the low four bits loaded to the Blue summing RAM, the next four bits loaded to the Green summing RAM and the next four bits loaded to the Red summing RAM. The high four bits of the color data word are ignored.

When accessing the summing RAMs, the data lines of the main lookup table are used as address lines as well as other lines as noted on the figure 12-2.

Loading the Monochrome Summing Rams

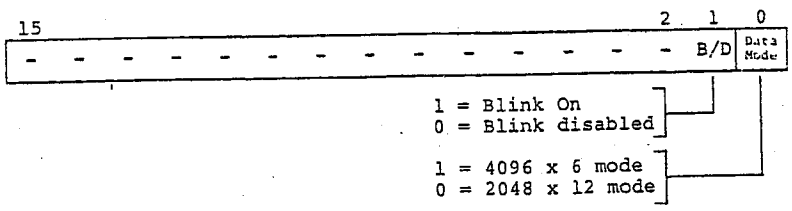
The monochrome summing RAM is a 256 X 8 lookup table located on memory Page 1 at memory locations 1200 hex through 13FE hex. Monochrome summing RAM data is loaded using the LCB command, with the following data format.



The low four bits of the monochrome data word are loaded to the selected address in the low order monochrome summing RAM (D0, D1, D2, D3) and the next four bits of the monochrome data word are loaded to the selected address in the high order monochrome summing RAM (D4, D5, D6, D7)

Loading the Command Status Register

The CSR selects the main lookup table memory mode and enables or disables the blink oscillator at A0 (low order address bit of main look up table). The CSR is loaded using the LCB command on Page 1 at address 1400 hex. The format of the CSR word is as follows:

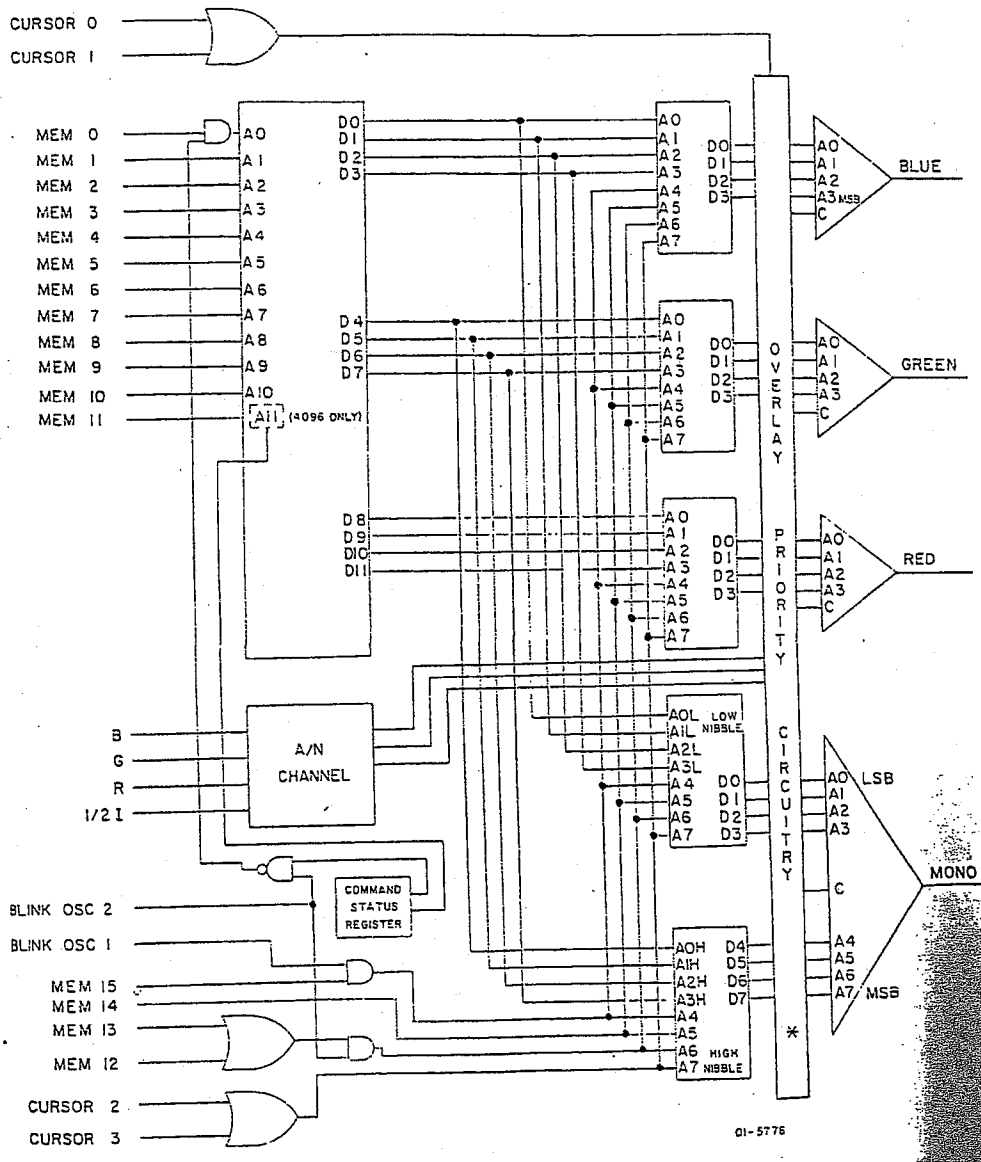


AX209869

Reading From the VID-101 Card

All lookup tables and the CSR may be read from 5216 memory by using the Read Cache Buffer (RCB) command for the appropriate addresses on memory Page 1.

Figure 12-2. VID-101 Wiring Diagram



* ORDER OF PRIORITY
1) CURSOR 0,1
2) A/N CHANNEL
3) REFRESH LOGIC

AX209870

SECTION 13

DATA TRANSMISSION TO HOST

The 5216 Standard Firmware provides several functions to enable data transmission to the host computer. The position of any of 15 cursor coordinates, the contents of refresh memory, of the video lookup table, and of the Cache Buffer may be requested by the host computer. Also, keystroke data typed on the 5115 keyboard may be transmitted to the host. The transmits are formatted to provide other pertinent data and to enable mirror transfers of data back to the 5216 as meaningful command sequences. Transmit formats are shown in Appendix A.

After any instruction requesting data transmission is issued, the host must immediately read all the data contained in the reply sequence. The 5216 will refuse to receive any more instructions until all data in the reply sequence has been read.

Transmit Cursor and Extended Transmit Cursor

The cursor location for any parameter may be transmitted to the host by using the TCO or XTCO instruction. The only difference between these two instructions is that in the data stream sent to the computer in response to TCO, the SMC is used while the response to XTCO includes the XSMC. XSMC is necessary for systems in which there are channels in the high four channel positions.

The low four bits of the TCO and XTCO instructions specify the parameter set from which the cursor location is to be transmitted (see SOM).

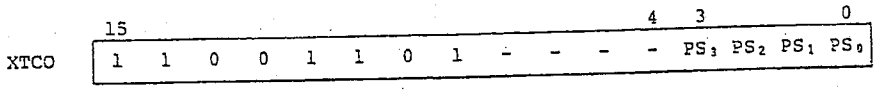
Cursor Position Readback - TCO

	15					10	9					4	3			0
TCO	1	0	0	0	1	1	-	-	-	-	-	-	PS_3	PS_2	PS_1	PS_0

After this instruction is sent, the cursor position stored in the parameter set specified in the low four bits of TCO will be sent to the host computer. See Appendix A for the reply sequence which will be sent to the 5216 in response to this instruction.

AX209871

Extended Cursor Position Readback - XTCO *EXT



The XTCO instruction causes the selected Display Generator to transmit the current contents of the cursor X and Y registers to the computer.

XTCO is functionally the same as TCO. However, in the data stream sent to the computer, XSMC is sent in response to XTCO while SMC is sent in response to TCO.

Transmit Picture Data

Refresh memory may be transmitted from the 5216 Display Computer to the host by using the Transmit Selected Channels (TSC), Extended TSC (XTSC) or Transmit Picture Elements (TPX) instructions. TSC and XTSC transmit data in Block Transfer (Word mode) one channel at a time within rectangular limits. TPX transmits data using the Block Transfer Mode (Pixel Mode), one pixel value at a time within rectangular limits.

Memory Data (TSC) and Extended Memory Data (XTSC) Readback

These two functions cause the 5216 to transmit refresh memory data to the host computer. The reply sequence is performed only for selected major and minor channels. Data is transmitted starting from the upper left corner of the rectangular limits, regardless of the cursor position, and proceeds left to right, top to bottom.

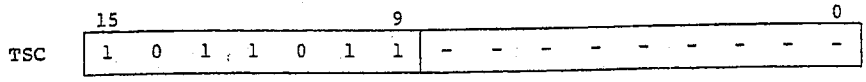
Because the data is transmitted one channel at a time, the transmission format contains the Select Major Channel (SMC), and Select Minor Channels (SMN) instructions, which select each channel sequentially. These sequential channel selects are embedded in the reply stream so that the data transmitted to the host can be sent back to the 5216 in the same form. The channels from which data will be transmitted may be limited by selecting major or minor channels before calling TSC or XTSC.

Data is transmitted in BXM graphic Word mode form regardless of the MCW status at the time the transmit instruction was issued.

TSC and XTSC are identical except that the reply format for TSC contains SMC and the reply format for XTSC contains XSMC. XSMC is necessary for systems in which there are channels in the high four channel positions.

AX209872

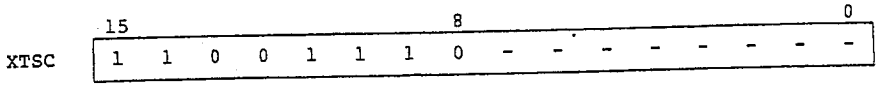
Memory Data Readback - TSC



The TSC instruction causes the Display Generator to transmit the data within the rectangular limits in each selected memory channel, one channel at a time. Data returned is graphic element information for each raster line of each memory channel.

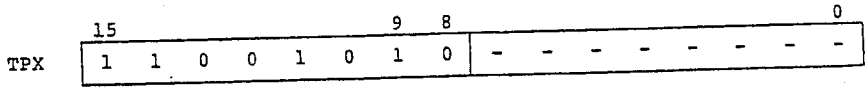
Display instructions are inserted where necessary so that the Display Generator output of memory data may be transmitted at a later time to recreate the displayed image originally read back. The data output format is shown in Appendix A.

Extended Memory Data Readback - XTSC *EXT



XTSC causes the Display Generator to transmit the data within the rectangular limits in each selected memory channel, one channel at a time. XTSC is functionally similar to TSC. However, in the data stream sent to the computer in response to XTSC, display instruction XSMC is inserted, while SMC is inserted in response to TSC.

Transmit Picture Elements - TPX *EXT



The Transmit Picture Elements (TPX) instruction causes all picture element information in selected channels within the rectangular limits to be transmitted to the host CPU. The transmit format is shown in Appendix A.

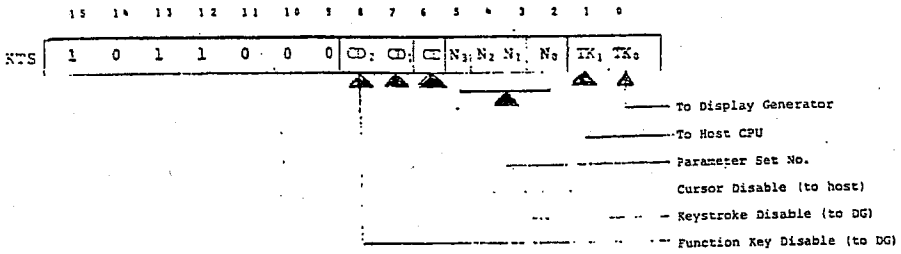
TPX uses the BXM in Pixel mode regardless of the MCW status at the time the transmit instruction was issued.

AX209873

Keyboard Transmit Status - KTS

The KTS instruction is provided for systems which use up to ten 5116 display keyboards with the 5216 and the host computer system.

The KTS instruction is used to select the destination of keystroke data.



The setting of the nine LSBs will affect the transmission path of all relevent keystroke/graph tablet/cursor data in the following manner.

Bit #	
8 7 6 1 Ø	
0 0 0 0 0	Locks out all keyboard functions except as noted below.
0 0 0 0 1	All keyboard functions acted on by DG.
0 0 0 1 0	All keyboard functions transmitted to host CPU.
0 0 0 1 1	All keyboard functions acted on by DG and transmitted to host CPU.
0 0 1 0 0	Locks out all keyboard functions except as noted below.
0 0 1 0 1	All keyboard functions acted on by DG.
0 0 1 1 0	All keyboard functions except cursor transmitted to host CPU.
0 0 1 1 1	All keyboard functions to DG cursor only; disabled to host CPU.
0 1 0 0 0	Locks out all keyboard functions except as noted below.
0 1 0 0 1	Cursor keys only; acted on by DG.
0 1 0 1 0	All keyboard functions transmitted to host CPU; no DG action.
0 1 0 1 1	All keyboard functions sent to host CPU; cursor only acted on by DG.
0 1 1 0 0	Locks out all keyboard functions except as noted below.
0 1 1 0 1	Cursor keys only; acted on by DG.
0 1 1 1 0	All keyboard function but cursor transmitted to host CPU.
0 1 1 1 1	All keys but cursor transmitted to host CPU and cursor only acted on by DG.
1 0 0 0 0	Locks out all keyboard functions except as noted below.
1 0 0 0 1	All keys but function keys acted on by DG.

1 0 0 1 0	All keyboard functions transmitted to host CPU.
1 0 0 1 1	All keys but functions acted on by DG. All keys transmitted to host CPU.
1 0 1 0 0	Locks out all keyboard functions except as noted below.
1 0 1 0 1	All keys but functions acted on by DG.
1 0 1 1 0	All keys but functions acted on by DG; all keys but cursor transmitted to host CPU.
1 0 1 1 1	All keys but functions acted on by DG; cursor only disabled to host CPU.
1 1 0 0 0	Locks out all keyboard functions except as noted below.
1 1 0 0 1	Cursor keys only acted on by DG.
1 1 0 1 0	All keys transmitted to host CPU; no DG action.
1 1 0 1 1	All keyboard functions transmitted to host CPU; cursor only acted on by DG.
1 1 1 0 0	Locks out all keyboard functions except as noted below.
1 1 1 0 1	Cursor keys only acted on by DG.
1 1 1 1 0	All keys but cursor transmitted to host CPU.
1 1 1 1 1	All keys but cursor transmitted to host CPU; cursor only acted on by DG.

Note

The Transmit Page and Transmit Cursor keys on the 5116 keyboard always load the output buffer to the host computer. On reception of the keyboard arming sequence, an interrupt to the host computer followed by a normal keystroke transmit sequence will occur. These two red keys are not affected by KTS.

When KTS specifies transmission from the keyboard to the host, the transmit format is as given in Appendix A. Key data is transmitted as ASCII codes in the low eight bits of the reply data. The high eight bits will be 0's. Each keystroke uses the seven-word transmit format.

The parameter set number associated with the keyboard from which the data is transmitted will be set in the low four bits of the SOM word contained in the reply stream.

When bit 6 is set, transmission of cursor movement keystrokes to the host is disabled. Setting bit 6 does not affect transmission of any other keystroke data. Bit 6 does not affect local operation of the keyboard cursor keys. The only function affected by setting this bit is the transmission of cursor movement keystrokes to the host computer.

When bit 7 of the KTS instruction = 1, the data output from the keyboard to the Display Generator is ignored except for cursor move instructions from the cursor move keys and the trackball or joystick.

AX209875

Bits 7 and 8 have higher priority than bit 0, so for example if bits 0 and 7 both are equal to 1, the only keys which will affect the 5216 are the cursor movement keys and the trackball or joystick.

When bit 8 is set to 1, the function keys on the 5116 keyboard will cause no action on the 5216 Display Computer. Bit 8 affects only the function keys.

Keyboard Transmission to Host

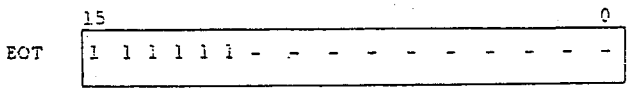
When the KTS instruction enables keyboard transmission to the host, every time a key is pressed on the 5116 keyboard, the appropriate key data is stored in an output buffer.

The host computer must signal the 5216 that is ready to receive this buffered key data by holding all three function lines high. The data is then sent across the hardware interface in a seven-word format as listed in Appendix A.

The keys numbered 0 through 10, found at the lower right of the 5116 keyboard, send two seven-word sequences to the host. The key data of the first sequence is 155 decimal (9B hex). The key data of the second sequence is the number engraved on the keytop.

In addition to these 10 special keys, an optional 45 key pad is available. These keys also return two seven-word sequences. The number associated with these 45 keys are shown in Appendix G.

End of Transmission - EOT



All data transmission from the 5216 to the host computer is automatically formatted by the 5216 to include an End of Transmission (EOT) word.

EOT signals the host that transmission from the 5216 is completed. (For more detailed information on handshaking between the host and the 5216, see Section 19.)

Note that the programmer never needs to send an EOT instruction for any reason. Inclusion of the EOT word at the end of the message is handled automatically by the firmware.

AX209876

SECTION 14

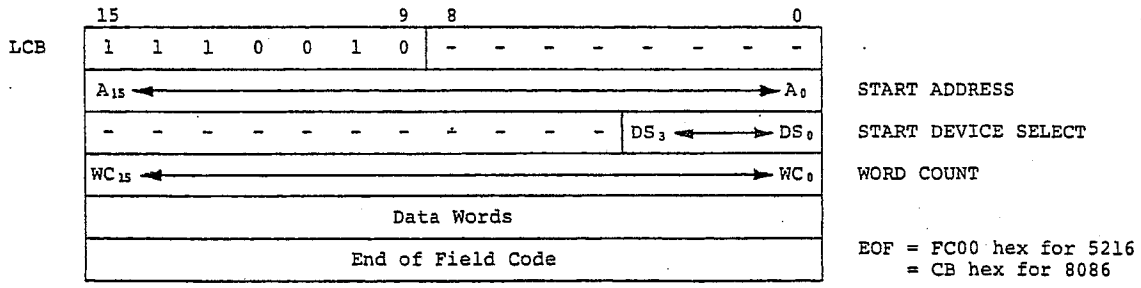
USER PROGRAMMABILITY

A block of RAM in the 5216 may be allocated in which sequences of commands may be stored by the user. These sequences may then be executed using a single 5216 instruction. The Cache Buffer may be loaded in two modes. The Display List (DL) mode allows the user to build files of frequently used 5216 command sequences which may then be executed by the Execute Cache Buffer (XCB) instruction.

The Writable Control Store (WCS) mode enables 8086 machine code routines to be stored and executed, thereby creating a new, user defined function for the 5216 Display Computer. Also, there is a Keyboard Buffer which may be loaded from the keyboard using character and function code keys.

The Cache Buffer must be loaded from the host but may be executed from the keyboard or the host. The Keyboard Buffer may be loaded from the host (using Load Cache Buffer) or from the keyboard and may be executed from either.

Load Cache Buffer - LCB *EXT



This instruction causes the 5216 to load user specified Cache Buffer commands starting at the address specified by:

DS_3 DS_2 DS_1 DS_0 A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0

This can be any address within the one megabyte addressing space of the machine.

The Device Select for the processor page is E hex (1110). The command does not report errors if there is no memory (or if the memory is ROM) at the addresses being loaded. Addresses E0000 to 7C00 are reserved for system operation. Addresses 7C02 to E7FFF are RAM and can generally be used for Cache and Keyboard Buffer storage. The Cache Buffer is written either in 5216 instructions or in 8086 machine code. If more memory is required, Expansion RAM modules are available. Expansion RAM is addressed using the Device Select for the individual system configuration. Expansion RAM addressing usually starts on Page 2.

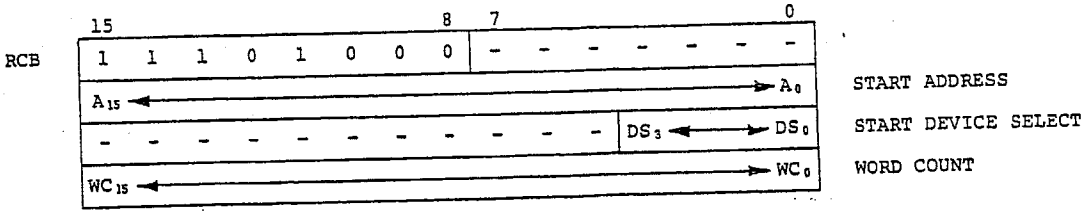
The last word of the Cache Buffer should be loaded with a terminating word. For the 5216 instruction Cache Buffer (Display List), the terminator is FC00 hex. For an 8086 machine code Cache Buffer (Writable Control Store), a long return (CB hex) should be entered in the terminating word. (The word count includes The End of Field word.) Certain instructions are not permitted in the Cache Buffer. See the paragraph describing (XCB) for a list of such instructions.

It is also possible to load the keyboard executable Keyboard Buffer using the LCB instruction. In this case, commands are bytes packed low byte, high byte, and the terminator byte is a 9B hex.

Since Load Cache Buffer is a direct memory load, it is also used to load the VID-101 card lookup tables.

*EXT

Read Cache Buffer - RCB



A Cache Buffer of a length specified by the word count is read back from the 5216 to the host (using the format shown in Appendix A) starting at the twenty-bit address defined by:

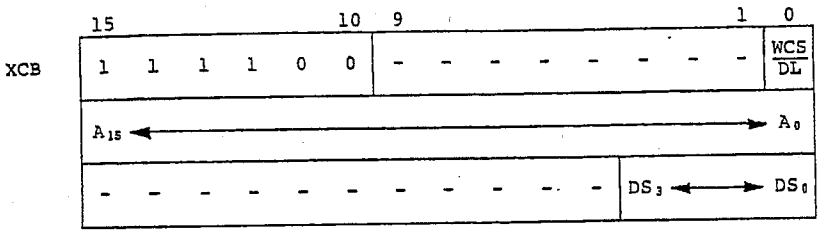
DS_3 DS_2 DS_1 DS_0 A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0

The RCB instruction does not check to see if memory or readable device registers are actually present at the locations being read back.

Notice that the data transmitted in response to the Read instruction includes an LCB instruction so that the host program can mirror the data back to the 5216 without any modification. (Refer to Appendix A.)

Since Read Cache Buffer is a direct memory Read, it is also used to read the contents of the VID-101 lookup tables.

Execute Cache Buffer - XCB *EXT



Word 1 Bit 0 0 = Display List (5216 Standard Firmware Instruction Set)
1 = Writable Control Store (8086 Machine Code)

The XCB instruction causes the 5216 to execute commands stored in the 5216 memory starting at the location defined by:

$DS_3\ DS_2\ DS_1\ DS_0\ A_{15}\ A_{14}\ A_{13}\ A_{12}\ A_{11}\ A_{10}\ A_9\ A_8\ A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$

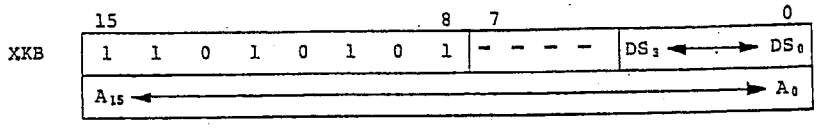
XCB can be used in two modes. In the first mode, Display List (with the WCS/DL bit set to 0), XCB fetches 16-bit display commands from sequential addresses and executes them one at a time unless an End of Transmission (EOT) command ($FC00_H$) is encountered. The following commands are not allowed from a Display List stored in a Cache Buffer:

LCB, RCB, TCO, XTCO, TSC, XTSC, BXM (Graphic), XBXM (Graphic), TPX, SDEV, RUL4, XRLU4, XLLU4.

These commands are not allowed from a Cache Buffer because they involve DMA transfers or other interaction with the host computer.

In the second mode, Writable Control Store (with the WCS/DL bit set to 1), the 5216 executes an intersegment subroutine call to the address specified. The 5216 processor then executes user supplied machine code until a user supplied intersegment return is executed (CB hex). The size of the Cache Buffer is limited only by the amount of memory contained in the 5216.

Execute Keyboard Buffer - XKB *EXT



This command executes a Keyboard Buffer starting at the location defined by:

DS_3 DS_2 DS_1 DS_0 A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0

Keyboard Buffer is a series of continuous bytes in 5216 memory which represent a series of ASCII keystrokes. Bytes are read from the buffer one at a time and executed by the Keystroke Handler program, using the current parameter set selected by the most recent Start of Message (SOM) instruction. Execution continues until a Terminate Keyboard Buffer code ($9B_H$) is encountered.

A user can set up a buffer from the keyboard, draw a picture, or write a message on the display using the keyboard and at the same time record keystrokes, terminate the buffer from the keyboard, transmit the buffer to the host using the RCB command, transmit it back to the 5216 at a later time using the LCB command, and redraw the picture or regenerate the message using the XKB command.

The on-board Device Select for the processor card is 0 and the off-board Device Select is E hex, so a Keyboard Buffer loaded from the host at E7C00 hex will be executed from the keyboard at 07C00 hex. To allocate more extensive memory than is available on the processor board, the optional Expansion RAM modules can be used.

The Keyboard Buffer can be loaded from the host with the Load Cache Buffer (LCB) instruction. Function code data corresponding to the appropriate function keys are loaded into memory (packed low byte, high byte), and terminated by the Terminate Keyboard Buffer (TKB) function code. This Keyboard Buffer can then be executed using the Execute Keyboard Buffer instruction from the host or the keyboard.

List Processing Instructions - Optional

A set of list processing instructions, formerly known as Standard List, is available with Standard Firmware. These instructions are most useful in conjunction with 5216 Standard Firmware dual processor systems.

In dual processor systems, one processor, the "host" processor, contains an operating system and higher level language and is used to load 5216 expansion RAM with lists of Standard Firmware instructions. The other processor contains Standard Firmware and executes instruction lists.

The list processing instructions enable the Standard Firmware processor to control list execution with more flexibility than simple beginning-to-end execution. They can perform conditional and unconditional jumps within a list, and repetitive loops. List loops can be nested within other loops.

The power of these instructions is not limited to dual processor systems; those with one processor can also effectively use list processing. Instruction lists can be downloaded from the host computer to 5216 expansion RAM, using the LOAD CACHE BUFFER instruction. They can then be executed by the 5216 by issuing just one instruction, EXECUTE CACHE BUFFER, from the host. Thus, the host computer and host interface are saved from long I/O time for loading often used instruction sequences.

Included with the list processing option is an interrupt-driven, automatic "Execute Cache Buffer" feature, which permits the programmer to use the "host" processor to store a list of display commands in 5216 memory and then send a signal to the standard list processor to begin execution of the display list. This process is performed by using a mailbox to store the address of the display list and an interrupt which causes the list processor to check the mailbox and begin executing the code stored at the address it finds there.

The mailbox is located at addresses E0000 and E0002. The page number or "device select" of the list is stored in address E0002 and the offset of the list address is stored in E0000. When the list processor encounters an end of list code (FC00) it resets the contents of E0000 to zero and then an interrupt is issued to the "host" processor. When the "host" processor receives Interrupt Vector 29, it vectors control to a "finish executing list" interrupt service routine whose address has been stored by the programmer at addresses F0076 (page of routine address) and F0074 (offset of routine address.)

Before executing the list the programmer must load the display list in 5216 memory, the display list address in the mailbox (E0000 and E0002), the "finished executing" service routine and the "finished executing" service routine address (at F0076 and F0074) and then the interrupt can be issued.

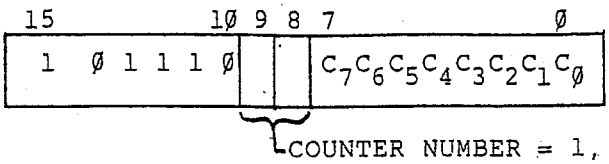
The interrupt is issued by writing a 2 to port 39 and then immediately a 0 to 39 to clear the port. The interrupt is latched into the programmable interrupt controller (PIC). For example, to execute a list display that starts at address 250FE the programmer loads "2000" into E0002 and "50FE" into E0000.

Before executing this list the address of the "finished executing" service routing must be loaded. For example, if the service routine is loaded at address 0700E, "0000" is loaded into F0076 and "700E" into F0074. Then to begin execution of the display list a "2" is written to port 39 and then a "0".

The display list starting at "250FE" is executed until "FC00" is encountered. The control is returned to the "host" processor and execution begins from address 0700E.

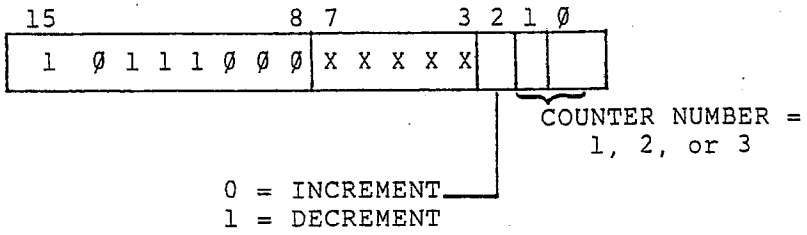
AX209881

SETCTR



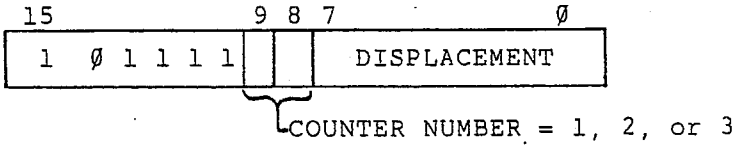
This instruction sets one of the three list counters to the 8-bit value given in the lower byte of the instruction. The counter numbers can be 1, 2, or 3. Zero is an illegal counter number.

INCCTR



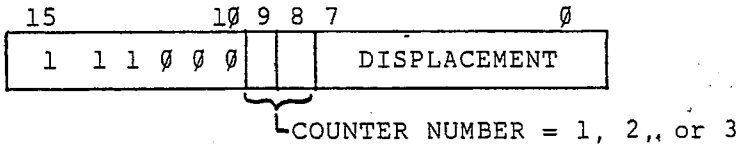
This instruction increments or decrements the list counter identified by the least significant two bits of the instruction. The counter value is incremented or decremented by one. Bit 2 of the instruction controls whether the counter is incremented or decremented.

LOOP

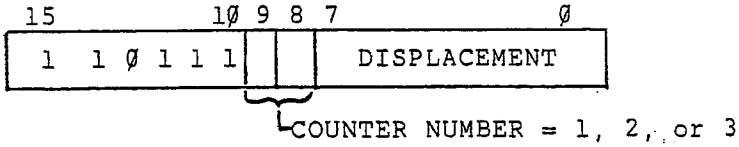


This instruction decrements the list counter specified by bits 8 and 9 of the instruction and tests for a zero result. If the list counter is decremented to zero, the 8-bit signed displacement in the low byte of the instruction is added to the current list pointer.

JMPZ

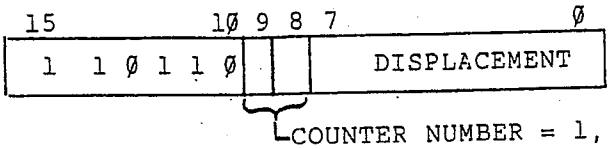


JMPPOS



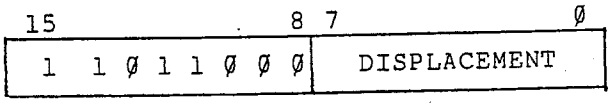
AX209882

JMPNEG



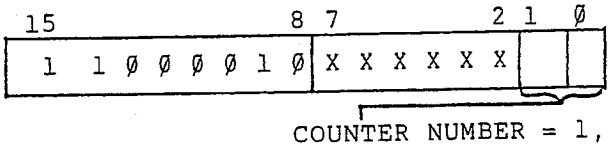
These three instructions test the 8-bit signed value of the designated list counter and add the 8-bit signed displacement to the list pointer if the test condition is met. JMPZ tests for list counter = Ø. JMPPOS tests for list counter positive (bit 7 of the list counter = Ø). JMPNEG tests for list counter negative (bit 7 of the list counter = 1).

JMPL



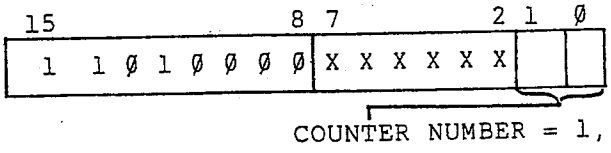
This instruction adds the 8-bit signed displacement given in the lower byte to the list pointer.

PSHLST



This instruction pushes the specified list counter onto its own counter stack. (NOTE: A 16-bit word is pushed onto the stack. The upper 8 bits of this word are undefined and the lower 8 bits are the list counter value.) The three counter stacks can contain a maximum of 16 words each. Attempts to overflow or underflow the stacks will be ignored.

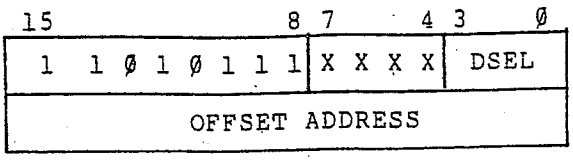
POPLST



This instruction pops a 16-bit word off the program stack and places the low eight bits in the specified list counter. This instruction and the PSHLST instruction may be used in combination to pass parameters from one list to another, or to machine language subroutines called from the list, or to copy a value from one counter to another.

AX209883

ALIST



This instruction sets the current list pointer to the 20-bit address specified in DSEL and offset address. DSEL contains the high order four bits and corresponds to the page number of the memory to be addressed. This instruction allows the program to jump from one list to another list anywhere in memory.

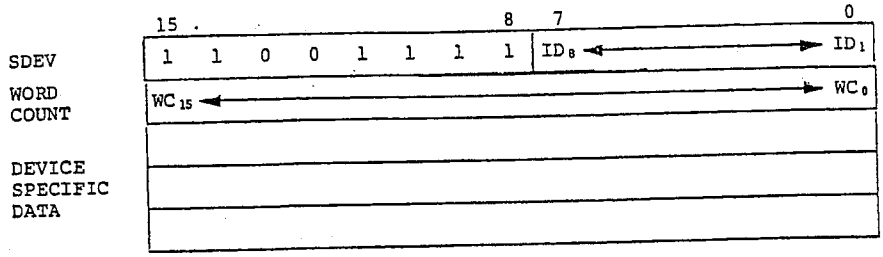
AX209884

SECTION 15

INTERNAL DEVICE ACCESS

Select Device - SDEV

*EXT



Host CPU control over 5216 devices (A/N Channels, disk, tape, controllers, etc) is provided by SDEV. The least significant eight bits select one of 256 devices, while the second word specifies a 16-bit word count. The following data, buffered when necessary, is down-loaded to the identified device. Consult your System Configuration Guide for device I.D. assignments.

Programming the Alphanumeric Channel Set using Select Device

Up to 15 ANCSs can be supported by the SDEV instruction. The ID number for the ANCS is 1 for the first ANCS, 2 for the second, etc.

The ANCS codes contained in the data field may be either command codes or character codes. To increase throughput, the data field is loaded into a 256-word buffer. Word transfers of greater than 256-word blocks may be specified, and will perform properly (such as during a full page load). However, such transfers will become stalled while the 256-word buffer is acted upon before further loading progresses. This will tie up the Display Generator DMA interface.

It is therefore recommended that the 256-word block size be used under normal circumstances. Since some instructions require longer times to perform (i.e., CLEAR), it is more efficient to use the interface for controlling other non-ANCS display activities, while the A/N channel is busy performing these lengthy operations.

Transmit Format for ANCS Transmit Instructions

There are several instructions in the ANCS firmware which cause transmission of ANCS data to the 5216 Processor Card (XMIT Line, XMIT Screen, XMIT Buffer and Read Cursor). When one of these transmit instructions is sent to the ANCS, the 5216 Standard Firmware will perform the necessary handshaking with the ANCS and send the data to the host computer. Data will be transmitted in blocks of 256 words or less as described above.

The format for data transmission from the 5216 to the host is: SOM word, a SDEV instruction, the data stream, and finally an End of Transmission instruction (EOT). The data words are 16-bit words which contain two 8-bit codes packed low byte, high byte.

When the Standard Firmware has data from the ANCS ready for transmission to the host, an Attention will be set, and DSTAT A and DSTAT B are set simultaneously. (Consult ANCS user documentation.)

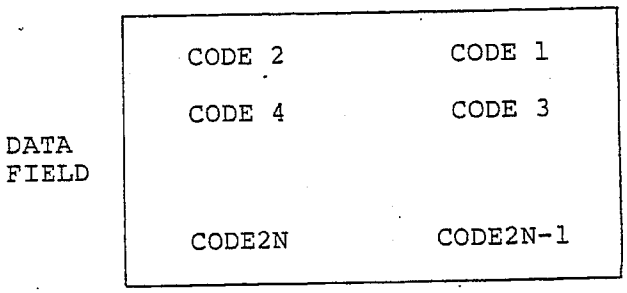


Figure 15-1. A/N Channel Block Transfer Format